Are you on Mobile or Desktop? On the Impact of End-User Device on Web QoE Inference from Encrypted Traffic

Sarah Wassermann*, Pedro Casas*, Zied Ben Houidi†, Alexis Huet†, Michael Seufert‡
Nikolas Wehner‡, Joshua Schüler‡, Shengming Cai†, Hao Shi†, Jinchun Xu†, Tobias Hossfeld‡, Dario Rossi†
*AIT Austrian Institute of Technology, †Huawei Technologies Co. Ltd, ‡University of Würzburg

***Abstract*—Web browsing is one of the key applications of the Internet, if not the most important one. We address the problem of Web Quality-of-Experience (QoE) monitoring from the ISP perspective, relying on in-network, passive measurements. As a proxy to Web QoE, we focus on the analysis of the well-known SpeedIndex (SI) metric. Given the lack of application-level-data visibility introduced by the wide adoption of end-to-end encryption, we resort to machine-learning models to infer the SI and the QoE level of individual web-page loading sessions, using as input only packet- and flow-level data. In this paper, we study the impact of different end-user device types (e.g., smartphone, desktop, tablet) on the performance of such models. Empirical evaluations on a large, multi-device, heterogeneous corpus of Web-QoE measurements for the most popular websites demonstrate that the proposed solution can infer the SI as well as estimate QoE ranges with high accuracy, using either packet-level or flow-level measurements. In addition, we show that the device type adds a strong bias to the feasibility of these Web-QoE models, putting into question the applicability of previously conceived approaches on single-device measurements. To improve the state of the art, we conceive cross-device generalizable models operating at both packet and flow levels, offering a feasible solution for Web-QoE monitoring in operational, multi-device networks. To the best of our knowledge, this is the first study tackling the analysis of Web QoE from encrypted network traffic in multi-device scenarios.**

I. Introduction

The Web is one of the most relevant components of the Internet. The performance of the Web is highly relevant to the success of every online service, as it severely impacts the engagement and churn of users. The assessment of a web service as perceived by the end user can be realized through the corresponding Web Quality of Experience (QoE), which is very challenging to measure. Different from other services, such as video streaming or gaming, web browsing is a composite of numerous multimedia components and embedded services; loading a web page today requires tens of flows to get the various page resources located in diverse servers from different content providers. In this complex process, the network plays a key role impacting users' Web QoE, forcing Internet Service Providers (ISPs) to deploy effective means to monitor Web QoE as perceived by their customers.

The literature on web performance analysis presents a wide range of objective metrics capturing the performance of web pages, including metrics such as Page Load Time (PLT), SpeedIndex (SI), and Above the Fold Time (AFT). However, all these metrics require access to the application layer, which is hidden from the eyes of the ISP by the wide deployment of end-to-end network traffic encryption.

The analysis of Web QoE from purely in-network, encrypted traffic measurements is yet an under-explored problem; in fact, we have been the first recently addressing it, **for the specific case of desktop web browsing** [1], [2], using **packet-level features** as input. By using controlled page-load experiments, where network data is simultaneously collected with ground-truth Web-QoE metrics such as SI and AFT, we have shown the potential of using supervised Machine Learning (ML) to infer these metrics from features computed on the encrypted stream of packets. In this paper, we follow a similar approach to [1], [2], extending the analysis in multiple new directions:

1 – Multi-device models: we consider Web QoE not only for desktop devices, but include smartphone and tablet Web QoE, conceiving cross-device generalizable models. The lion's share of Internet-access devices today is smartphones, with nearly three quarters of the world population using exclusively their smartphones to access the Internet by 2025 [3]. As we find in our results, a model trained only on desktop browsing data provides poor Web-QoE estimation performance when applied to smartphone and tablet measurements.

2 – Web-QoE Estimation: besides training regression models to estimate Web-QoE objective metrics – in particular SI –, we rely on real end-user data from previous studies [4] to build Web-QoE classification models for subjective metrics – e.g., Mean Opinion (MOS) Scores.

3 – ML Benchmark: we present an extensive benchmark comparing the performance of different ML models for both estimation tasks – regression for SI inference, and classification for QoE estimation.

4 – Packet and Flow-level Models: we conceive models working either at the packet-level or at the flow-level; we show that the proposed flow-level models achieve highly similar performance to the packet-level ones, but using an order of magnitude less input features, thus showing strong potential for a practical monitoring solution.

The remainder of the paper is organized as follows. Section II overviews the related work on Web-QoE monitoring and analysis. Section III presents the overall modeling and data-generation approach, including a characterization of the produced datasets for this study. In Section IV, we introduce

and evaluate the proposed packet-level ML models for two different tasks, including Web-QoE inference (SI) and classification of QoE ranges (excellent, good, poor); here we also show how models trained on single-device measurements – e.g., desktop –, result in strong performance degradation when applied to other device types such as smartphones and tablets. In Section V we present a (per-task) unified, multi-device model, which achieves high inference/prediction performance with a strong generalization potential across desktop, smartphone, and tablet devices. In this section we also extend the multi-device models to operate at the flow-level, showing that the same performance can be achieved using much less, and easier to compute, input features. Finally, Section VI concludes this paper. As a conclusion, the proposed multi-device, flow-level models lay the basis for a generalizable, multi-device, Web-QoE passive monitoring system.

II. Related Work

Initial Web-QoE models were based on Page Load Times (PLT) [5], [6], and are still broadly used in the practice to infer user satisfaction in web browsing, e.g, under ITU-T [7]. However, research has demonstrated that PLT is a poor proxy to user perception of web-page loading times. Indeed, the actual web content visible to the user is usually displayed much earlier, as most web pages often stretch beyond the browser's viewport. Additional in-browser metrics have been accordingly devised to better suit the page display on the screen. An approach is the so-called Above the Fold Time (AFT), i.e., the time until the visible portion of a web page has been fully loaded, which has been also tested within traditional Web-QoE models [8]. Newer Web-QoE metrics have been proposed recently, such as the SI, which considers the whole visual progress of the page loading, by processing a video capture of the screen. Besides single metric modeling, ML-based approaches have been explored [4], [9] to model Web QoE from a combination of metrics.

Another direction in the literature proposes to understand how external components influence Web QoE. Prior work [10], [11] has studied the impact of network-quality fluctuations and outages on user Web QoE. But besides network quality, other components influence Web QoE. They are linked to the specific web-page content – usability [12], aesthetics [13], etc. – as well as device type: desktop, smartphone, tablets [14]. Important to our study, these papers show that smartphones and tablets have their own characteristics, not only regarding screen sizes, but also in terms of content rendering and web designs. Most of these papers focus on Web QoE in controlled, small-scale lab environments.

Others directly rely on in-browser metrics as a proxy to infer Web QoE, conducting large-scale active measurement campaigns. For example, the impact of multiple features such as transport protocols and network performance on PLT and AFT is studied in [15], based on a set of 244 million measurements collected during 6 months for the top-10000 Alexa websites. Other papers also measured the impact of similar features on

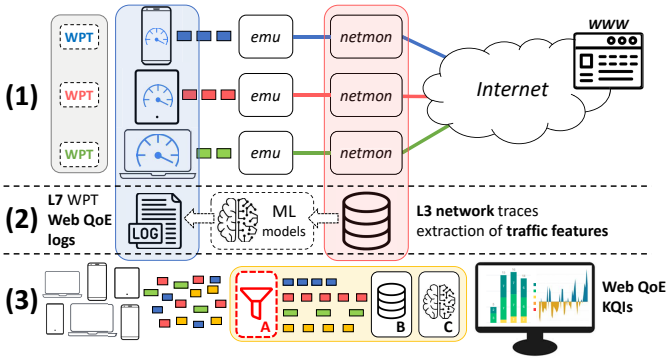


Fig. 1: Diagram and workflow of the proposed solution.

PLT and SI or AFT in different countries and different types of networks [16], including mobile ones [17].

Most of prior work has stayed at the application level, which is problematic for ISPs, who have no direct access to in-browser metrics. In recent years, TLS encryption has even narrowed the information that ISPs can collect from the network side, and previous approaches [18] based on deep packet inspection (DPI) and HTTP-traffic analysis are no longer applicable. Other papers [19], [20] developed *correlated approximations* to the SI metric, such as Byte/Object-Index [19] and Pain-Index [20], which can be computed from statistics of packet- and flow-level measurements, thus seamlessly operating with encrypted traffic. In recent work [1], [2], we took a step further to directly infer the SI metric, using ML techniques mapping network (encrypted) packet-level traffic features to SI, in desktop devices.

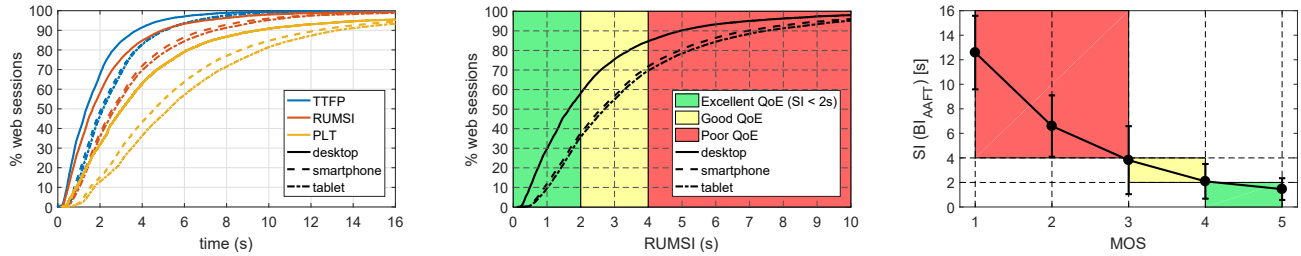
When it comes to the specific case of Web-QoE monitoring in mobile devices, there have been multiple papers using machine learning [21]–[24] or simple modeling approaches [25] to map application-layer metrics [23], [24] or network-QoS metrics [21], [22], [25] into QoE-related metrics. From these, two papers [21], [22] are the closest to our work, but both propose analysis approaches which are no longer applicable due to HTTP-traffic encryption [22], or do not address the specific problem of web browsing [21].

In this paper, we deal with an unexplored and so far neglected issue: understanding the impact of the end-user device type on the analysis of Web QoE from in-network traffic measurements. To the best of our knowledge, this is the first paper addressing this challenge.

III. Web-QoE Datasets & Modeling Approach

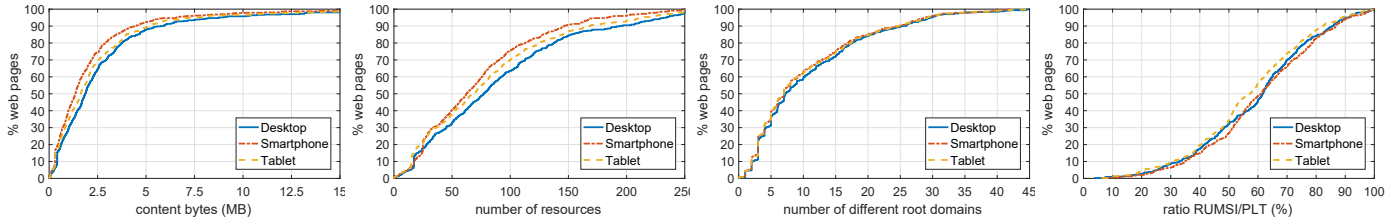
The proposed solution to the Web-QoE monitoring problem consists of training supervised ML models to map network-traffic features, extracted from the encrypted network-web-page traffic, into relevant Web-QoE metrics. The approach is data-driven, and thus needs datasets containing both the collected traffic traces – the *input* – and the targeted Web-QoE metric – the *ground truth*. The diagram presented in Figure 1 presents the workflow and the different stages of the solution.

To fully control the generation of such datasets, we built a measurement testbed based on multiple private instances of



(a) Time performance metrics per device. (b) RUMSI and QoE classes. (c) SI vs. MOS scores.

Fig. 2: Distribution of (a) TTFP, RUMSI, and PLT values, (b) QoE classes per device type, based on (c) real-user MOS scores.

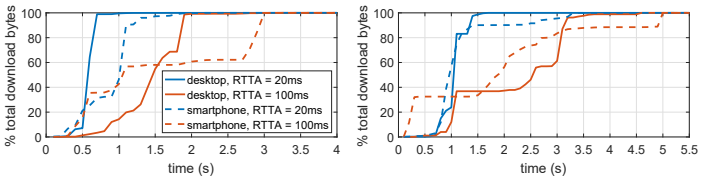


(a) Bytes. (b) Resources. (c) Root domains. (d) RUMSI/PLT ratio.

Fig. 3: Data characterization, per device type, including page size, number of resources, number of root domains, and SI/PLT ratio. The latter reflects the complexity of the web page in terms of visible content (SI) and full content downloading (PLT).

WebPageTest (WPT) – https://www.webpagetest.org/, the default, open-source web-performance-analysis tool used both in industry and academia. Figure 1, rows **(1)** and **(2)** describe this testbed and its usage. Different from previous studies [1], [2], [15]–[17], [19], which have studied Web QoE exclusively for desktop browsers and desktop devices (or in some exceptional cases, emulating mobile devices), our measurement testbed consists of three different, non-emulated types of devices, including a smartphone device (Google Pixel 2 XL), a tablet (Google Pixel Slate), and a desktop computer (laptop), using WPT agents for Android and Linux. Chrome (last stable version) is used as browser. Instead of leveraging in-device WPT traffic-shaping capabilities, devices are connected to the open Internet through independent network emulators (*emu*), which allows for more realistic network-access-performance configurations in terms of bandwidth, latency, packet-loss rate, etc. This allows for heterogeneity in the generated measurements. Configurations used in the study include access-downlink bandwidth up to 10 Mbps, packet-loss rates up to 10%, and RTTs up to 100 ms. Using WPT measurements, the platform extracts about 90 different KPIs and Web-QoE metrics, indicated as L7 Web-QoE logs in Figure 1, row **(2)**, such as PLT, SI, AFT, Byte Index [19], and Time to Interactive (TTI), as well as content characteristics of the visited pages. Network traffic is captured at an intermediate passive monitoring device (*netmon*) and stored as .pcap traces, from where model input features are extracted, indicated as L3 network-traffic features in Figure 1, row **(2)**. For this study, we generated a per-device-type balanced dataset of more than 50.000 web page *loading sessions* (i.e., the loading of a single page), targeting the top 500 websites according to the Alexa top-sites list (https://www.alexa.com/topsites). The same pages are visited multiple times for each device type, using the same access-network setups. We do not consider the effect of caching, i.e., tests correspond to a first-view loading session. As we have recently shown in [1], while caching has an impact on the performance of inference models, this impact is limited, and models generalize well across different protocol and caching settings [1]. We focus on the inference of one particular Web-QoE metric, the SI, which is today one of the most accepted metrics reflecting Web QoE. Nevertheless, the methodology applies to any other similar Web-QoE metric. As shown in [19], measuring the SI is cumbersome in terms of computational resources and might introduce bias in the data capturing/processing, mainly due to the video capturing and analysis. This is particularly critical on smartphones and tablets, which are generally resource-constrained; therefore, instead of focusing on the SI metric, we collect the so-called RUM SpeedIndex (RUMSI) metric [26], which is a passive approximation to the SI, computed from the analysis of web-page resource timings.

To conclude, note that we assume that the measurement system takes as input network traffic from single web sessions. In an operational deployment in the wild – see row **(3)** in Figure 1 –, the traffic mix of concurrent web sessions must first be disentangled (step **A**) to then extract features from the traffic belonging to each web session (step **B**) and apply the trained model(s) (step **C**). This paper exclusively addresses steps **B** and **C**. Nevertheless, in case of concurrent web sessions, a classification methodology from the literature [20] could be applied to disentangle them. While the Web-traffic identification/disentangling (step **A**) is out of the scope of this paper, we have conceived multiple techniques addressing this problem.



(a) https://zoom.us (b) https://cnn.com

Fig. 4: Examples of CBD features or loading curves, using ΔT = 100 ms, and different Access-RTT (RTTA) setups.

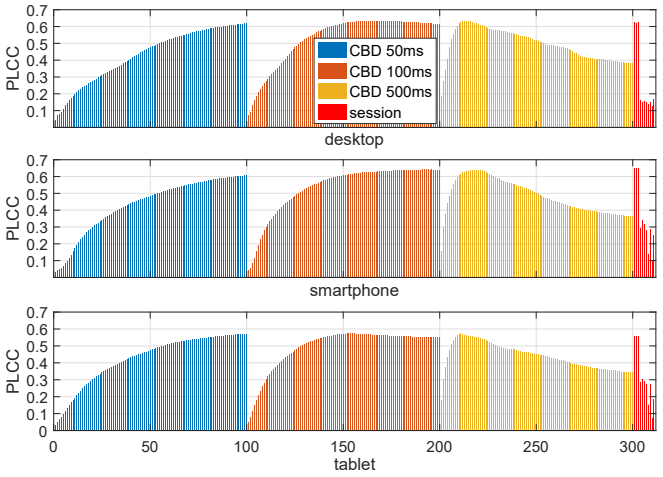


Fig. 5: Correlation between input features and RUMSI.

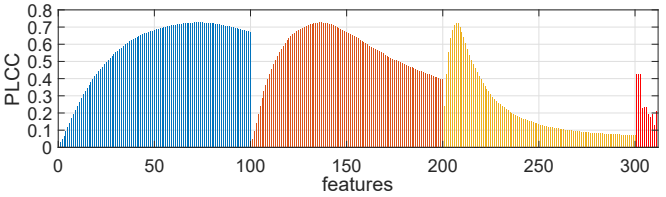


Fig. 6: Correlation between input features and QoE class.

A. Data Characterization

The list of top 500 Alexa pages is assorted in terms of contents, and as we show next, the type of device being used has a visible impact on web-page characteristics and timing performance. Figure 2(a) depicts the distribution of three relevant Web-QoE metrics, including the Time to First Paint (TTFP), which accounts for the time at which the first object is painted on the browser, the RUMSI, and the PLT. Note how the values are significantly higher for both smartphone and tablet devices as compared to desktop devices, pointing to a more complex rendering process in mobile devices. This is most probably linked to the specific hardware limitations of smartphones and tablets, as well as the particular characteristics of the OS and browser combination – native Chrome in Android. In addition, the way pages are optimized (most times dynamically) and rendered in mobile devices impacts loading times. Worse loading performance in mobile devices is a commonly known issue in practice; see page-speed statistics at https://backlinko.com/page-speed-stats. Interestingly, when comparing Android devices, TTFP values are almost identical for smartphone and tablet, RUMSI is slightly higher for tablet, while PLT is significantly higher for tablet. As we see next, this is most probably explained by the fact that web pages have more content to load in tablets. It is also interesting to note how PLT significantly overestimates the perceived loading time of web pages, represented by the (RUM)SI metric.

Figure 3 characterizes the 500 web pages per device type, in terms of (a) page size, (b) number of resources, (c) number of root domains, and (d) RUMSI to PLT ratio. The latter reflects the complexity of the web page in terms of visible content (SI) and full content downloading (PLT). As expected, web pages browsed in desktop devices are bigger than those browsed on smartphones or tablets, which are optimized for smaller screen sizes. The average page size is 2.7MB in desktop, 2.4MB in tablet, and 2.1MB in smartphone. Figures 3(b) and 3(c) further illustrate the richness and complexity of the web pages in terms of number of embedded contents and their location at different root domains, with more than 30% to 35% of the web pages consisting of more than 100 resources, and about 40% of the web pages fetching resources from more than 10 different root domains. The screen size probably plays a key role in terms of page characteristics, as the number of resources is higher for desktop, followed by tablet, and finally by smartphone. The RUMSI/PLT ratio shows not only how large the overestimation introduced by PLT is in terms of perceived page load times, but also how different this is for the different web pages. Indeed, about 10% of the pages have a ratio below 0.3 (the visible content loads way faster than the full content) and less than 5% of the pages have a ratio above 0.9 (the visible content basically corresponds to the full web page content).

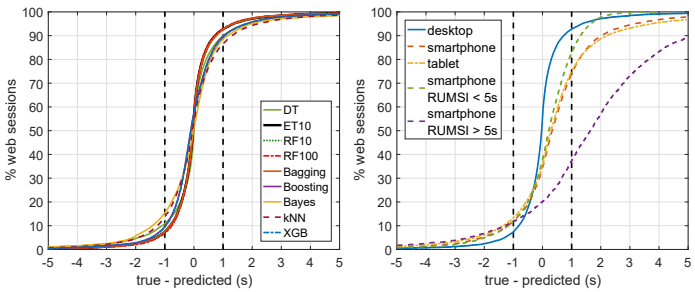
Conclusion: while most of the analyzed pages are very similar for every device type in terms of size and retrieved contents from external servers, differences can be significant for a small share of the pages. In terms of performance, loading times in Android devices (smartphone and tablet) are significantly higher than in desktop, a common trend observed in practice of web page speed analysis.

B. Subjective QoE Analysis

While the SI is a good objective metric reasonably capturing the Web QoE of real users [8], we resort to previous subjective Web-QoE studies to better understand the expected QoE for the generated dataset. In particular, prior work [4] conducted a subjective study where about 240 participants rated their browsing experience – loading of individual pages using a desktop browser –, according to a 5-level Absolute Category Rating (ACR) MOS score (*bad* QoE being 1 to *excellent* being 5). We rely on their publicly available dataset to identify QoE-related timing thresholds which could translate the RUMSI in our dataset to broad QoE classes. In Figure 2(c) we depict the relationship between SI and MOS scores obtained in that study. While the SI metric was not directly measured, additional metrics such as the Byte Index where computed,

model	MAE-mAE (ms)	MRE-mRE (%)	PLCC
DT	766 – 288	37 – 18	0.817
ET10	**598 – 260**	**31 – 16**	**0.879**
RF10	602 – 262	31 – 16	0.860
RF100	**564 – 249**	**30 – 15**	**0.885**
Bagging	600 – 266	31 – 16	0.857
Boosting	767 – 426	48 – 26	0.861
Bayes	976 – 491	64 – 29	0.727
kNN	940 – 496	54 – 29	0.752
XGB	774 – 429	48 – 26	0.849

TABLE I: Benchmarking of different ML models for RUMSI inference, for desktop.



(a) Desktop. (b) Cross-device testing.

Fig. 7: RUMSI inference performance. Models are trained using exclusively desktop measurements.

which can be used as a good proxy to the real SI [19]. Based on these subjective QoE results, we define 3 Web QoE classes: **(e)**xcellent – MOS $\geq$ 4 –, **(g)**ood – 3 $\leq$ MOS $<$ 4 –, and **(p)**oor – MOS $<$ 3 –, resulting in SI thresholds of 2 and 4 seconds. Interestingly, the SI thresholds recommended in the industry as target for excellent Web performance vary between 1 second (desktop) and 3 seconds (mobile), which are in line with the proposed higher QoE class threshold of 2 seconds. It is important to note that our thresholds are derived for the case of browsing on desktop devices, and one would expect higher thresholds for Web QoE in mobile devices. Nevertheless, for this study, we assume the same thresholds apply to the three device types. Figure 2(b) shows that about 60%/40% of the loading sessions correspond to excellent QoE in desktop/mobile (smartphone and tablet) respectively, 25%/30% to good QoE, and the remaining 15%/30% result in poor QoE.

C. Targets and Input Features

We realize the Web-QoE monitoring solution through two different prediction tasks: (i) inference of the RUMSI metric, which corresponds to a regression task, and (ii) prediction of the Web-QoE class {e,g,p}, which corresponds to a 3-class classification task. We use the same input features in both tasks, derived from the stream of encrypted packets. To define input features, we follow the rationale behind the computation of the SI metric itself, which considers the whole progress of the page loading. We define the Cumulative Bytes Downloaded features $CBD(i)_{\Delta T}$, as the (normalized) cumulative number of bytes downloaded from the first collected byte at time t_0 up to time $t = t_0 + i \times \Delta T$, with $i = 1, \ldots, m$. The CBD features track the download progress of the page bytes, using a time resolution ΔT. Figure 4 depicts examples of CBD features for different network configurations, both for desktop and smartphone devices, using $m = 100$ and $\Delta T = 100$ ms. Pages loading faster have a CBD *loading curve* rising sharper and arriving to full loading earlier.

For this study, we take $m = 100$ samples, and three different resolutions for the computation of features, using ΔT = 50 ms, 100 ms, and 500 ms, for a total of 300 CBD features. Using different resolutions helps capture different phenomena in the downloading progress, which potentially impact the SI, as well as allowing to track different page-load durations, in this case up to 5, 10, and 50 seconds, respectively. We consider $n = 11$ additional input *session* features, related to the complete page-loading session; these include: full session duration (first to last packet), downlink/uplink session duration (first to last packet in downlink/uplink direction), total number of packets downlink/uplink/full, total number of bytes downlink/uplink/full, and session mean throughput downlink/uplink. While these are mostly packet-level features, we extend in Section V-A the study to the implementation of flow-level features, realizing highly similar results.

Figure 5 depicts the linear correlation between these input features and the RUMSI metric, for different device types. Correlation values are rather high for all devices, with stronger correlations observed for CBD features between 5 seconds and 10 seconds, as well as for session-duration features. Figure 6 shows correlation values for the QoE classification problem, for all devices together; as expected, based on the considered time-thresholds, higher correlations are observed between 2 and 5 seconds.

IV. Desktop Models' Lack of Generalization

We now focus on the Web-QoE estimation tasks. As a reference for performance evaluation, we begin by training and evaluating different ML models for the specific case of desktop measurements. Recall that desktop measurements represent the most common data source so far used in the Web-QoE literature [1], [2], [15]–[17], [19]. We then apply the trained models to both smartphone and tablet data, to highlight the cross-device lack of generalization and poor performance realized by single-device models in such multi-device scenarios. As a general note on the evaluations in this paper, all performance results correspond to 5-fold cross validation.

A. RUMSI Inference on Desktop

Table I reports the RUMSI inference performance achieved by nine different ML models, most of them based on decision trees. The tested models include single decision tree (DT), multiple types of ensembles using different numbers of trees, such as extremely randomized trees (ET), random forest (RF),

model	desktop						
	ACC	R{e}	R{g}	R{p}	P{e}	P{g}	P{p}
DT	80.3	88.8	66.1	72.8	88.4	66.2	73.8
ET10	**84.4**	**93.1**	**70.5**	**75.6**	**89.6**	**73.7**	**81.7**
RF10	84.6	93.1	71.6	74.6	90.1	73.1	82.1
RF100	**86.9**	**93.3**	**77.4**	**79.1**	**92.3**	**76.2**	**84.7**
Bagging	85.7	93.2	74.3	76.8	90.8	74.6	84.8
Boosting	82.9	91.1	70.3	73.7	90.1	69.1	79.3
Bayes	60.1	93.0	11.6	19.5	63.2	38.3	46.7
kNN	74.9	87.3	55.1	62.1	81.8	59.1	72.0
XGB	82.2	90.9	69.0	72.1	89.8	67.9	77.6

TABLE II: Benchmarking of different ML models for Web-QoE prediction, on desktop. The three levels of QoE correspond to **excellent**{**e**}, **good**{**g**}, and **poor**{**p**} QoE.

device	MAE-mAE (ms)	MRE-mRE (%)	PLCC
desktop	598 – 260	31 – 16	0.879
smartphone	1245 – 721	41 – 28	0.728
tablet	1434 – 724	44 – 28	0.618
smartphone RUMSI $<$ 5s	867 – 592	43 – 29	0.455
smartphone RUMSI $>$ 5s	2812 – 2000	32 – 27	0.667

TABLE III: Inference performance per device type. The ET10 model is trained using desktop data.

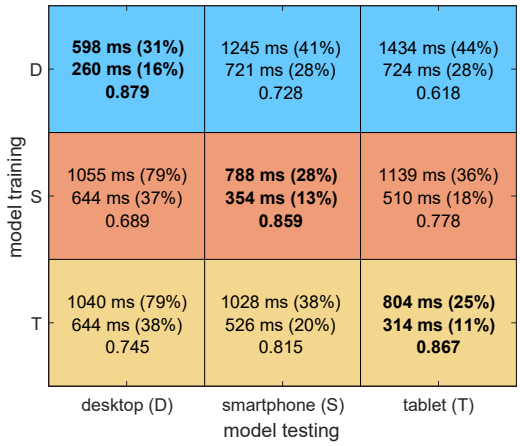


Fig. 8: Cross-device inference performance, using per-device ET10 as underlying model.

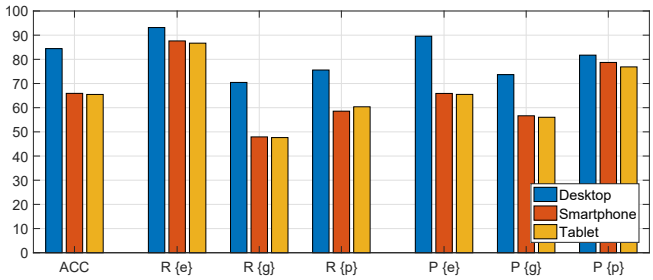


Fig. 9: Cross-device QoE classification performance. Models are trained on desktop measurements.

bagging trees, and boosting including XGBoost [27]. The list is completed by a plain Bayesian approach, and by the k nearest neighbors algorithm (kNN). We assess performance using 3 performance metrics for regression problems, including the absolute error (AE), the relative error (RE), and the linear correlation (PLCC). We take both mean (M) and median (m) values for the error metrics, to filter out significantly large errors. Figure 7(a) additionally depicts the distribution of the prediction errors.

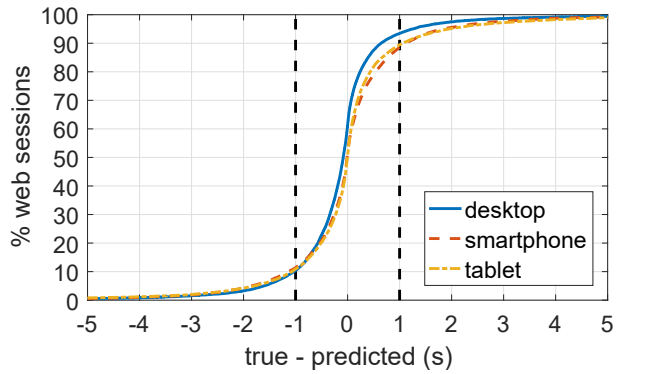
RF100 attains the best inference performance, with a median absolute error of 249 ms, and a median relative error of 15%. Absolute prediction errors are below 500 ms for more than 70% of the sessions. More than 85% of the RUMSI values are inferred with an error below one second. Similar performance is realized by smaller ensembles, e.g., RF10, ET10, and bagging, using 10 instead of 100 trees. Given the training-speed improvements obtained with the ET10 model, we take it as the underlying prediction model in subsequent evaluations.

B. QoE Classification on Desktop

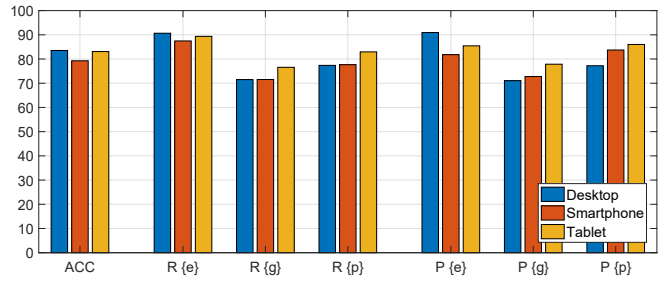
Table II reports the classification benchmark results obtained for desktop. Again, RF100 provides the best results, with an overall accuracy (ACC) close to 87%. Recall (R) and precision (P) are above 90% for the excellent QoE class prediction, but good and poor performance classes tend to be confused by the predictor. Nevertheless, recall and precision are close to 80% for these QoE classes.

C. Lack of Generalization for Mobile Devices

Now that we have built the models for desktop, a natural question is: *how good would these models perform in data collected from other device types?* This is critical in practice, as a significant, and ever growing, share of web-browsing activity comes from mobile devices. Figure 7(b) depicts the distribution of inference errors per device type, using the ET10 model, trained exclusively on desktop data. Table III summarizes the corresponding performance metrics. There is a strong inference-performance degradation when applying the desktop model to both smartphone and tablet data. Median absolute errors almost triple as compared to desktop performance. The distribution of errors shows that the desktop model tends to underestimate the RUMSI metric when applied to other devices, not present at training time. One could argue that RUMSI on smartphone and tablet is significantly higher than on desktop (cf. Figure 2), thus the performance degradation could be linked exclusively to this mismatch. However, performance degradation is also significant when only considering smaller RUMSI values, with median errors



(a) RUMSI Inference.



(b) QoE Classification.

Fig. 10: (a) RUMSI inference and (b) QoE classification performance, ET10 model training on all-device data.

more than doubling for the example case of smartphone – from 260 ms to 592 ms –, testing only for RUMSI below 5 seconds.

Figure 8 shows how the aforementioned cross-device training and validation issues also hold when considering different device types, using the RUMSI inference as example. The figure reports the usual AE, RE and PLCC metrics arranged as a training/testing matrix, where rows correspond to the device-type data used for training, and columns to the device-type data used for testing. While specialization improves inference performance – the matrix diagonal –, training a model on measurements from a particular device type and applying the resulting model on measurements from a different device type results in poor inference performance, for all device-type combinations. Note also how the cross-device lack of generalization applies to mobile devices, which are closer in terms of characteristics (cf. Figure 2 and Figure 3); while the performance degradation is lower when considering cross-data from smartphone/tablet devices, it is still non-negligible.

Performance degradation is also significant for the QoE classification problem. Figure 9 reports the classification performance per device type, again using the ET10 model, trained on desktop data. Overall classification accuracy drops from 84% on desktop to 67% on smartphone and tablet. Recall for excellent QoE degrades only slightly, but strongly for the other classes, and most importantly, precision for excellent QoE also drops strongly, meaning that the model cannot correctly track the classification problem.

device	MAE-mAE (ms)	MRE-mRE (%)	PLCC
desktop	649 – 300	37 – 18	0.874
smartphone	804 – 363	27 – 14	0.855
tablet	798 – 318	25 – 11	0.868
all	750 – 327	30 – 14	0.869

TABLE IV: Multi-device RUMSI inference performance.

Conclusion: RUMSI inference and QoE prediction can be properly realized using CBD and session-based features, extracted directly from the stream of encrypted bytes. However, models so far proposed in the literature for single device types [1], [2], [15]–[17], [19] might not perform properly in the wild, where other devices than desktop machines are used for web browsing.

V. Multi-device (Flow) Web QoE Models

Having shown the lack of generalization and the cross-device issues introduced by per-device models, we take the most natural step to conceive multi-device Web-QoE models. Possible approaches include the usage of stacking/ensembles of specialized models [28], or the inclusion of a pre-processing device-type classification task, preceding the main inference/prediction task. However, the simplest approach, exposing models to all devices data at training time, already provides high accuracy and generalizes well across devices, which has high practical appeal as it simplifies deployment.

Considering multi-device RUMSI inference first, Table IV and Figure 10(a) summarize the performance attained by a single ET10 model, trained on all-device data. Compared to per-device specialized models (cf. the matrix diagonal in Figure 8), there is a marginal degradation for the corresponding multi-device model, and mainly observed for desktop, with an error increase close to 10%. Still, performance is consistent across the three device types, with an overall median absolute error of 327 ms, and a relative error of 14%. Overall, the generalization capabilities of the multi-device model outweigh the accuracy of the specialized models, making it a preferred choice for Web-QoE monitoring in operational deployments.

Considering multi-device QoE classification next, Figure 10(b) reports the performance obtained with a single ET10 model, trained on all-device data: again, a slight performance degradation compared to the specialized desktop model (cf. Table II), but yields significant gain in terms of generalization to mobile devices (cf. Figure 9). The overall model accuracy is 82.2%, with recall and precision values for **excellent**{**e**}, **good**{**g**}, and **poor**{**p**} QoE of about $\{89\%, 73\%, 80\%\}$ and $\{86\%, 74\%, 82\%\}$, respectively.

Conclusion: multi-device models significantly improve generalization of the Web-QoE inference across different devices, with only slight under-performance as compared to specialized models. As such, multi-device models provide simple, more accurate, and more reliable monitoring capabilities in realistic web-browsing scenarios.

A. Multi-device, Flow-level Models

In the last part of the study we focus on improving the practical application of the proposed Web-QoE inference models. In particular, we explore the definition of new input features at the flow level, which could be easier to compute than the proposed packet-level features so far considered. We define a set of 21 flow-level features, using similar notions to the ones which guided the packet-level features. These include: (i) the total number of flows (all, downlink, uplink), (ii) the min/mean/median/max flow duration in downlink, (iii) the min/mean/median/max flow size in downlink, (iv) the min/mean/median/max flow byte-index in downlink, (v) the mean/median in-flow, average intra-packets time (MDT) in downlink, (vi) the mean/median/max flow throughput in downlink, and (vii) the Flow-Index (FI).

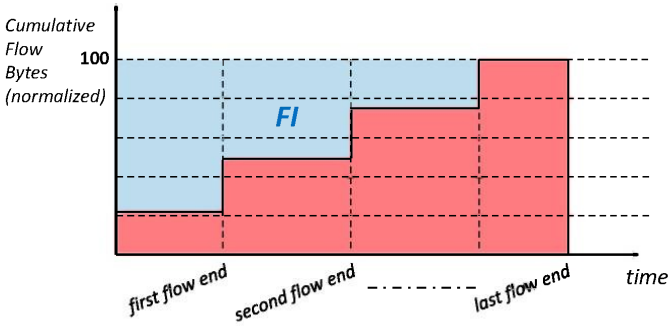


Fig. 11: Flow-level features – flow index.

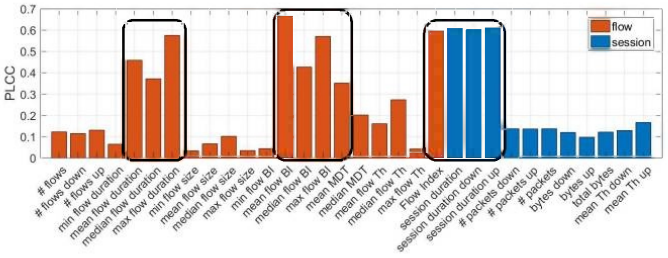


Fig. 12: Flow-level features, ranked by PLCC.

The flow byte-index uses the standard definition of Byte Index (BI) [19], but considering only the packets belonging to a specific flow. The FI feature represents an extension to the BI, but using flow size and flow ending time instead of packet size and time. Figure 11 depicts the basic notions behind the calculation of the FI. Both the FI and BI are integral-like metrics, similar to the definition of the SpeedIndex. We refer the reader to [19] for a comprehensive definition of the BI and the concepts of integral metrics. Flow features are complemented by the 11 session-level features, previously defined in Section III-C, adding to a total of 32 input features, computable at the flow level. For the sake of completeness, Figure 12 reports the linear correlation between these flow and session input features and the RUMSI metric. Features related to flow duration, flow BI, FI, and session duration are the ones showing the highest correlation to the RUMSI.

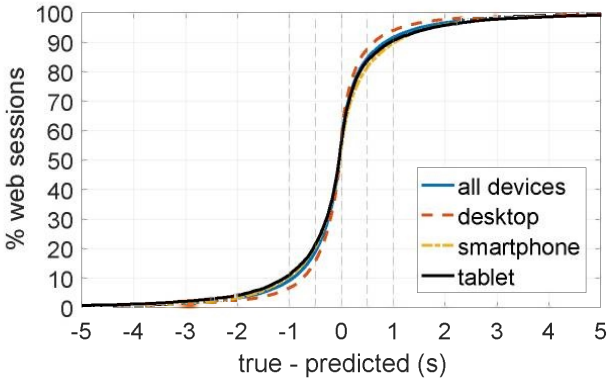


Fig. 13: Multi-device RUMSI inference performance – flow-level features.

device	MAE-mAE (ms)	MRE-mRE (%)
desktop	628 – 309	35 – 18
smartphone	815 – 364	26 – 13
tablet	751 – 317	25 – 11
all	732 – 324	29 – 13

TABLE V: Multi-device, flow-level RUMSI inference.

Figure 13 and Table V report and summarize the RUMSI inference performance achieved by a multi-device model, using as input the new set of 32 features. Results are comparable, and even slightly better for some device types, than those achieved by using packet-level features (cf. Table IV), with the paramount advantage of using an order of magnitude less features, and at an easier-to-compute and more scalable level.

VI. Concluding Remarks

We have tackled the problem of Web-QoE monitoring from the ISP perspective, relying on in-network, passive measurements. Empirical evaluations on a large, multi-device, heterogeneous corpus of Web-QoE measurements for the most popular websites shows that the proposed solution can infer the (RUM)SI as well as estimate Web-QoE ranges from in-network traffic measurements with high accuracy. At the same time, the device type introduces a strong bias in the capabilities of Web-QoE inference models, causing models trained for single device types to badly generalize to other devices. We showed that this cross-device lack of generalization can be solved by properly training on data coming from a multitude of devices. Our findings raise awareness of the fact that models for Web-QoE monitoring must be exposed to multi-device measurements to achieve proper inference and prediction performance in real network deployments, something generally neglected in the literature. To the best of our knowledge, we are the first to unveil the strong impact that the device type has on such models. The definition of novel flow-level features which also realize highly accurate predictions is also a key-contribution of the study.

REFERENCES

[1] A. Huet, et al., "Revealing QoE of Web Users from Encrypted Network Traffic," in *IFIP Networking Conference*, 2020.

[2] A. Huet, et al., "Web Quality of Experience from Encrypted Packets," in *ACM SIGCOMM Posters and Demos*, 2019.

[3] Cisco, "Cisco Annual Internet Report (2018-2023) White Paper, Updated March 2020," Cisco, Tech. Rep., 2020.

[4] D. N. da Hora, et al., "Narrowing the gap between QoS metrics and Web QoE using Above-the-fold metrics," in *PAM*, 2018.

[5] E. Ibarrola, et al., "Web QoE evaluation in multi-agent networks: Validation of ITU-T G.1030," in *ICAS*, 2009.

[6] S. Egger, et al., "Waiting Times in Quality of Experience for Web Based Services," in *QoMEX*, 2012.

[7] "G.1030 : Estimating End-to-end Performance in IP Networks for Data Applications," https://www.itu.int/rec/T-REC-G.1030.

[8] T. Hoßfeld, et al., "Speed Index: Relating the Industrial Standard for User Perceived Web Performance to Web QoE," in *QoMEX*, 2018.

[9] Q. Gao, et al., "Perceived Performance of Top Retail Webpages in the Wild: Insights from Large-scale Crowdsourcing of Above-the-fold QoE," in *Internet-QoE*, 2017.

[10] A. Sackl, et al., "The influence of network quality fluctuations on web qoe," in *QoMEX*, 2014.

[11] A. Sackl, et al., "Quantifying the impact of network bandwidth fluctuations and outages on web qoe," in *QoMEX*, 2015.

[12] M. Varela, et al., "QoE in the Web: A dance of design and performance," in *QoMEX*, 2015.

[13] M. Varela, et al., "Towards an understanding of visual appeal in website design," in *QoMEX*, 2013.

[14] S. Baraković, et al., "Survey of research on Quality of Experience modelling for web browsing," *Quality and User Experience*, vol. 2, no. 1, p. 6, 2017.

[15] A. Saverimoutou, et al., "A 6-month Analysis of Factors Impacting Web Browsing Quality for QoE Prediction," *Computer Networks*, vol. 164, 2019.

[16] A. S. Asrese, et al., "Measuring Web Latency and Rendering Performance: Method, Tools, and Longitudinal Dataset," *IEEE TNSM*, 2019.

[17] M. Rajiullah, et al., "Web Experience in Mobile Networks: Lessons from Two Million Page Visits," in *WWW*, 2019.

[18] S. Ihm, et al., "Towards Understanding Modern Web Traffic," in *ACM IMC*, 2011.

[19] E. Bocchi, et al., "Measuring the Quality of Experience of Web Users," *ACM SIGCOMM CCR*, vol. 46, no. 4, 2016.

[20] M. Trevisan, et al., "PAIN: A Passive Web performance indicator for ISPs," *Computer Networks*, vol. 149, 2019.

[21] V. Aggarwal, et al., "Prometheus: Toward Quality-of-Experience Estimation for Mobile Apps from Passive Network Measurements," in *ACM HotMobile*, 2014.

[22] A. Balachandran, et al., "Modeling Web Quality of Experience on Cellular Networks," in *ACM MobiCom*, 2015.

[23] P. Casas, et al., "Next to You: Monitoring Quality of Experience in Cellular Networks From the End-Devices," *IEEE TNSM*, 2016.

[24] S. Wassermann, et al., "Machine Learning Models for YouTube QoE and User Engagement Prediction in Smartphones," *SIGMETRICS Perform. Eval. Rev.*, 2019.

[25] A. Nikravesh, et al., "QoE Inference and Improvement Without End-Host Control," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018.

[26] P. Meenan, "Real User Monitoring SpeedIndex (RUMSI)," 2020. [Online]. Available: https://github.com/WPO-Foundation/RUM-SpeedIndex

[27] T. Chen, et al., "XGBoost: A Scalable Tree Boosting System," in *KDD*, 2016.

[28] P. Casas, et al., "GML Learning, A Generic Machine Learning Model for Network Measurements Analysis", in *CNSM*, 2017.